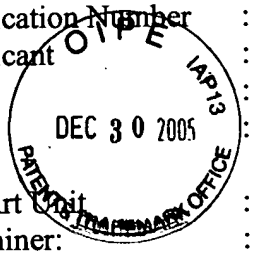
PATENT
Attorney Docket No. 53470.003034

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE



Application Number	:	09/883,499	Confirmation No.: 8688
Applicant	:	Jeffrey A. Bedell et al.	
Filed	:	June 19, 2001	
Title	:	SYSTEM AND METHOD FOR EFFICIENT DATA RETRIEVAL AND PROCESSING	
TC/Art Unit	:	2161	
Examiner:	:	Haythim J. ALAUBAIDI	
Docket No.	:	53470.003034	
Customer No.	:	21967	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

SUBMISSION OF APPEAL BRIEF

Appellant attaches hereto an Appeal Brief in connection with the above-captioned patent application. Pursuant to 37 C.F.R. § 1.192, three copies of the Appeal Brief are being submitted. The appeal fee of $500.00 (for a small entity) is included with the check for $2,020.00 (submitted herewith that includes the extension of time fees and notice of appeal fees). Any deficiency in or overpayment of this fee should be charged or credited to Deposit Account No. 50-0206

Respectfully submitted,

December 30, 2005

Brian M. Buroker
Registration No. 39,125

Hunton & Williams
1900 K. St., NW
Washington, D.C. 20006-1109
(202) 955-1894

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Number	: 09/883,499	Confirmation No.:	8688
Applicant	: Jeffrey A. Bedell et al.		
Filed	: June 19, 2001		
Title	: SYSTEM AND METHOD FOR EFFICIENT DATA RETRIEVAL AND PROCESSING		
TC/Art Unit	: 2161		
Examiner:	: Haythim J. ALAUBAIDI		
Docket No.	: 53470.003034		
Customer No.	: 21967		

APPEAL BRIEF

TABLE OF CONTENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Number	: 09/883,499	Confirmation No.:	8688
Applicant	: Jeffrey A. Bedell et al.		
Filed	: June 19, 2001		
Title	: SYSTEM AND METHOD FOR EFFICIENT DATA RETRIEVAL AND PROCESSING		
TC/Art Unit	: 2161		
Examiner:	: Haythim J. ALAUBAIDI		
Docket No.	: 53470.003034		
Customer No.	: 21967		

APPEAL BRIEF

In response to the Office Action dated July 15, 2005, finally rejecting pending claims 1, 2, 5-11, 14-18 and 21, appellant respectfully requests that the Board of Patent Appeals and Interferences reconsider and withdraw the rejections of record, and allow the pending claims, which are attached hereto as an Appendix.

I. Real Party In Interest

The real party in interest is Microstrategy, Incorporated as assignee of the entire interest in the above-referenced application, assigned by its inventors Jeffrey A. Bedell, Michael Codini, William Hurwood, Ashutosh K. Jhaveri, Benjamin Z. Li, Fabrice Martin, Sadanand Sahasrabudhe, and Jun Yuan.

II. Related Appeals And Interferences

There are no known related appeals.

III. Status Of Claims

Claims 1-3, 5-12, 14-21 and 26-29 are pending in this application. Claims 1, 2, 5-11, 14-18 and 21 stand rejected. Claims 3 and 12 stand objected to. Claims 19-20 and 26-29 stand allowed.

The rejection of claims 1, 2, 5-11, 14-18 and 21 is appealed.

IV. Status Of Amendments

No amendments to the claims have been filed subsequent to the final rejection dated July 15, 2005.

V. Summary Of Invention

Appellant believes that a brief discussion of the background technology, followed by a brief summary of the embodiments of the invention and the problems solved by the embodiments of the present invention, will assist the Board of Patent Appeals and Interferences (hereinafter referred to as "the Board") in appreciating the significant advances made by the embodiments of the present invention.

A. The Background

The ability to act quickly and decisively in today's increasingly competitive marketplace is critical to the success of any organization. The volume of data that is available to organizations is rapidly increasing, frequently overwhelming, and presents various challenges. One challenge is to avoid inundating an individual with unnecessary information. Another challenge is to ensure all relevant information is available in a timely manner.

One known approach to addressing these and other challenges is known as data warehousing. Data warehouses, relational databases, and data marts are becoming important elements of many information delivery systems because they provide a central location where a reconciled version of data extracted from a wide variety of operational systems may be stored. A data warehouse typically allows users to tap into a business's vast store of operational data to track and respond to business trends that facilitate forecasting and planning efforts.

Decision support systems have been developed to efficiently retrieve selected information from data warehouses. One type of decision support system is known as an on-line analytical processing system ("OLAP"). In general, OLAP systems analyze the data from a number of different perspectives and support complex analyses against large input data sets. A given OLAP system may handle interactions with a variety of database management systems (DBMS) simultaneously (such as when a single data warehouse includes multiple databases and DBMS from multiple vendors) or as a matter of compatibility with multiple competing DBMS.

OLAP systems may be used to retrieve and process data from large data sets, such as Very Large Databases (VLDBs). Very large data sets may contain a considerable amount of redundant data. In some cases, the redundant data may exist in varying levels of detail, aggregation, abstraction, or transformation (e.g., through formula or other calculation). As a result, there are frequently multiple ways to retrieve the same data set from a given data source. Additionally, many query languages contain a number of functions, shortcuts, and processing structures that may provide redundant methods of retrieving a given data set from the same set of tables.

Prior OLAP systems do not make efficient use of OLAP concepts and DBMS resources and capabilities. Prior OLAP systems also do not utilize knowledge of database schemas, DBMS resources and capabilities, intermediate data sets, and/or other properties of OLAP systems and VLDBs to efficiently retrieve and process data.

B. The Embodiments Of The Present Invention

Accordingly, the embodiments of the invention may overcome these and other drawbacks of prior OLAP systems. Specifically, the embodiments of the invention promote optimization of query language statements to be run against a data source. Even minor improvements in the

selection of base tables, intermediate data handling, function and calculation selection, and the use of DBMS specific syntax, shortcuts, and enhancements may dramatically improve the speed and processing efficiency of complex queries against large data sources.

One aspect of the embodiments of the invention is a system for retrieval and processing of a data set from a data source. The system includes a query structure assembly module, a syntax assembly module, and a process optimization module. The query structure assembly module defines a query structure based upon query assembly rules and a desired data set (such as the set of data for a particular OLAP report). The syntax assembly module defines one or more query language statements based upon the defined query structure. The process optimization module evaluates processing options based upon a database schema associated with the data source. One or more query language statements are assembled by the system and run against the data source to return the desired data set.

Another aspect of the embodiments of the invention is a method of generating a query language statement to be run against a data source. The method may include the steps of generating a query structure, generating query language syntax, and evaluating the structure and syntax for process optimization.

Still another aspect of the embodiments of the invention is a medium having a processor readable program code embodied therein for retrieving and processing data from a data source. The medium may include code for causing the processor to evaluate multiple sets of base tables within the data source for generating a desired data set. The medium may also include code for causing the processor to evaluate one or more intermediate tables for reusability in generating the desired data set. The medium may also include code for causing the processor to evaluate multiple methods for generating intermediate data for use in generating the desired data set. The

medium may include code for causing the processor to evaluate multiple join paths used for joining tables to return the desired data set. Finally, the medium may include code for causing the processor to assemble at least one query language statement based upon evaluation of the sets of base tables, the intermediate data tables, the methods for generating temporary tables, and the join paths.

C. Explanation of Independent Claim 1

A computer-implemented system (100) for retrieval and processing of a data set from one or more data sources comprises a query structure assembly module (106/312) for defining a query structure based upon a plurality of query assembly rules (312) and a desired data set (310), the query assembly rules being used by the query structure assembly module to evaluate the desired data set. *See, e.g.,* Spec. at pg. 6, lines 9-22 and pg. 13, lines 14-26. A syntax assembly module (118/314) defines at least one query language statement based upon the defined query structure. *See, e.g.,* Spec. at pg. 9, lines 5-19 and pg. 13, lines 1-10. Further, a process optimization module (316) evaluates processing options based upon a database schema associated with the data source (310), the process optimization module including an intermediate data processing method module (322) for evaluating a plurality of methods for generating intermediate data sets within the data source(s). *See, e.g.,* Spec. at pg. 14, lines 4-25. At least one query language statement is assembled and run against the data source(s) to return the desired data result set. *See, e.g.,* Spec. at pg. 8, lines 22-28 and pg. 9, lines 1-4, 15-23.

D. Explanation of Independent Claim 10

A computer-implemented method of generating a query language statement to be run against one or more data sources (200) comprises generating a query structure based upon a database schema associated with the data source, query assembly rules, and a desired data result

set, the query assembly rules being used to evaluate the desired data set. *See, e.g.,* Spec. at pg. 6, lines 9-22 and pg. 13, lines 14-26. The method also involves a step of generating query language syntax (404) based upon the query structure (402) for returning the desired data result set (226) from the data source(s). *See, e.g.,* Spec. at pg. 9, lines 5-19 and pg. 13, lines 1-10. Further, the method also involves evaluating a plurality of query assembly rules (502), syntax descriptions (504), or syntax patterns (506) for process optimization. *See, e.g.,* Spec. at pg. 14, lines 4-25. Last, the method involves evaluating a plurality of methods (606) for generating intermediate data sets. *See id.*

E. Explanation of Independent Claim 21

A computer-implemented system for constructing a structured query language statement to be run against at least one database comprises a query structure assembly module (312) for constructing a query structure based upon an evaluation of a desired data set by at least one query assembly rule (328). *See, e.g.,* Spec. at pg. 6, lines 9-22 and pg. 13, lines 14-26. The system includes a syntax assembly module (314) for defining at least one query language statement based upon the constructed query structure. *See, e.g.,* Spec. at pg. 9, lines 5-19 and pg. 13, lines 1-10. Further, the system includes a process optimization module (316) for evaluating the construction of the query structure and the defining of the at least one query language statement, the evaluation occurring during the construction of the query structure and/or during the defining of the at least one query statement, the process optimization module serving to evaluate a plurality of methods for generating intermediate data sets. *See, e.g.,* Spec. at pg. 14, lines 4-25.

VI. Grounds Of Rejection To Be Reviewed On Appeal

The following grounds of rejection are to be reviewed on appeal:

1) The rejection under 35 U.S.C. § 103(a) of claims 1, 6-10, 15-18 and 21 based on U.S. Patent No. 5,918,232 to Pouschine ("Pouschine") in view of U.S. Patent No. 5,584,024 to Shwartz ("Shwartz").

2) The rejection under 35 U.S.C. § 103(a) of claims 2, 5, 11 and 14 over Pouschine in view of U. S. Patent No. 6,574,623 to Leung ("Leung").

VII. Argument

The rejections of the claims in this case exemplify classic hindsight reconstruction that is contrary to the law. Controlling Federal Circuit and Board precedent require that the Office Action set forth specific and particularized motivation for one of ordinary skill in the art to modify a primary reference to achieve a claimed invention. *Ruiz v. A.B. Chance Co.*, 234 F.3d 654, 664 (Fed. Cir. 2000) ("[t]o prevent a hindsight-based obviousness analysis, [the Federal Circuit has] clearly established that the relevant inquiry for determining the scope and content of the prior art is whether there is a reason, suggestion, or motivation in the prior art or elsewhere that would have led one of ordinary skill in the art to combine the references."). Here, the Office Action cobbles together two or three references to allegedly yield the claims based on unsupported allegations of what is "conventional." These references all relate to different features without any specific teaching to be modified or combined into the claims presented by applicant. Then, for each dependent claim, the Office Action selects from amongst the two or three as though that were all that was required. For each such additional modification to the primary reference, the Office has the burden to establish motivation for that additional modification as well.

Simply put, the Office has failed to set forth a *prima facie* case of obviousness for any of the independent claims. Additionally, the Oficce has also failed to establish a *prima facie* case of

obviousness for the further modifications proposed to yield the dependent claims as well. Each of the specific claims and the impropriety of the rejections is addressed below.

A. The Rejection Of Claims 1, 6-10, 15-18, And 21 As Being Unpatentable Over Pouschine In View of Shwartz Under 35 U.S.C. 103(a) Is Improper

The Office Action rejects pending claims 1, 6-10, 15-18, and 21 for obviousness based on Pouschine in view of either Shwartz or Leung. Appellant respectfully submits that the cited references, alone or in combination, do not teach or suggest all the elements in the claimed invention.

On page 6 of the Office Action the Examiner references the following passages from Pouschine as allegedly disclosing "*a query structure assembly module based on query rules*":

> The method includes the steps of inputting a query, parsing the query; creating a query component tree, inputting model metadata into the Domain Modeling Rule Set Preparation Module, generating a Domain Modeling Rule Set from the Domain Modeling Rule Set Preparation Module, inputting the Domain Modeling Rule Set into the calculation engine, and generating an execution tree with rules from the calculation engine. (Col. 4, lines 61-67)
>
> The method 200 starts as a Hyperstructure Query Language (HQL) query 202 which is passed to a parser 122 which converts the HQL text to a query component tree 204 which represents the component parts of the query 202. (Col. 16, lines 23-26)

Appellant respectfully submits that the quoted passages (or the rest of the Pouschine reference for that matter) fail to show a query structure assemble module based on "query

assembly rules." It should be noted that Pouschine does not use the term "rule" in the same sense as in the present application. In the present application, "query assembly rules" are typically rules for evaluating the desired data set and for generating an optimized query execution plan. Page 13, lines 14 - 17. The query assembly rules may embody functions of the base selection module 318 (page 15, lines 11-13), the intermediate table selection module 320 (page 16, lines 3-5), the intermediate table method module 322 (page 16, lines 22-23), and/or the join path selection module 324 (page 17, lines 22-23), for example. That is, the query assembly rules may comprise a number of logical rules, dependencies and conditions which are utilized by the query structure assembly module 312 during initial parsing of the desired data set parameters and assembly of the query structure. Page 15, lines 9-15. In other words, the query assembly rules are rules that govern the query structure assembly process.

In Pouschine, the term "rule" is explained in the following passages:

> Model--A named, file-like package consisting of:
>
> * Dimensions, elements, and rules
> * Blobs to store third party data and worksheets
> * A set of user entered values
> * Information spaces (Col. 8, lines 1-5)
>
> To obtain the values for cells in the model, DOLAP uses two types of rules: 1) formula rules which tell DOLAP to compute the cell value mathematically from other cells in the model, for example, Revenue is calculated by multiplying Price times Units Sold; and 2) data map rules which tell DOLAP to retrieve the cell value from the database. (Col. 12, lines 1-6)

> Domain Modeling Rule Set Preparation is the process of gathering all of the rules from the model, turning them into valid actions that can be used by the calculation engine, and prioritizing them so that the proper rules take precedence. (Col. 17, lines 11-14)

See also, col. 12, lines 9-50, for a more detailed description of "formula rules" and "data map rules." At least two conclusions can be readily drawn from the above-quoted passages: (1) the rules in Pouschine come from user-defined models; and (2) these rules are either formulae for computing the cell value or data maps for retrieving the cell value. In other words, Pouschine's "rules" are essentially the query structure itself, rather than rules that govern the assembly of the query structure. Therefore, despite the confusing use of a similar word, Pouschine does not disclose "query assembly rules" as recited in claims 1, 10 and 21.

The Office Action acknowledges that Pouschine "does not explicitly indicate in the first limitation of the current claim [Claim 1] the step of basing the defining of a query structure on a plurality of query assembly rules." Office Action, page 8. However, the Examiner points to the following passage as allegedly disclosing this limitation:

> These are provided to Domain Modeling Rule Set Preparation 208 which generates the domain modeling rule set 126, which are then supplied to the calculation engine 18, which takes the applicable rules and adds them to the query tree 204 to produce an execution tree with rules 212 for the query engine 132. (Col. 16, lines 30-35, emphasis the Examiner's)

As discussed above, the "Domain Modeling Rule Set" in Pouschine is essentially user-defined formulae and/or actions for computing or retrieving the cell value. Since the "[calculation engine 18] takes the applicable rules and adds them to the query tree 204," the Domain Modeling Rule Set is part of the query structure itself. Therefore, the Domain Modeling Rule Set cannot be the functional equivalent of the "query assembly rules" that govern the assembly of query structure.

The Examiner, on page 6 of the Office Action, also quotes the following passages from Pouschine as allegedly disclosing the limitation of "*the query assembly rules being used by the query structure assembly module to evaluate the desired data set*":

> a Domain Modeling Rule Set Preparation Module, a query engine, and an evaluator which communicates with an SQL generator ... (Col. 4, lines 57-58, emphasis the Examiner's)

The underlining of "Rule Set" confirms that there may be some confusion about the meaning of "Rule Set" in the Pouschine reference and the "query assembly rules" in the present application.

Further, the "evaluator" in Pouschine does not "evaluate the desired data set" as claimed in the present application. Rather, the evaluator 128 "decides whether further data is required from a relational database. If further data is required, evaluator 128 communicates with a Relational DataBase Management System (RDBMS) 216 through an SQL generator 218. Evaluator 128 can also communicate with a math library 220, if a calculation is required, or a sorting and processing system 222, if the process requires ordering of results or sorting in some manner." Col. 16, lines 38-46. As such, Pouschine's evaluator 128 is more like an intermediary between the query engine and the relational database. See Figure 8.

The Examiner further quotes on page 7 of the Office Action the following passages from Pouschine as allegedly disclosing "*a syntax assembly module for defining at least one query language statement*":

> The method 200 starts as a Hyperstructure Query Language (HQL) query 202 which is passed to a parser 122 which converts the HQL text to a query component tree 204 which represents the component parts of the query 202. (Col. 16, lines 23-26)

However, the quoted passage only pertains to the *parsing* of a HQL text but does not teach the *assembling* of a query structure.

Also on page 7 of the Office Action, the Examiner references the following passages from Pouschine as allegedly disclosing "*a process optimization module for evaluating processing options based upon a database schema*":

> An incoming client request is routed to an HQL parser 122 which transforms the HQL query from its textual form into an internal representation. After the internal representation of the query is made, the calculation engine 18 looks in the query results cache 124 to see if the query (or any portion of it) can be retrieved from the cache in order to save processing time.
>
> A Domain Modeling Rule Set 126 receives input from both the model 50 and the schema 104, and provides input to the rule evaluator 128, which then communicates with the execution plan 130 and the query engine 132. The query engine 132 creates SQL queries from HQL queries and optimizes these SQL queries by combining them (whenever possible)

> and also optimizes the performance of the calculation engine 18 by performing as many calculations as possible with SQL statements executed by the database. (Col. 15, lines 51-66)
>
> This is followed by inputting the execution tree with rules to the query engine, generating an optimized execution tree from the query engine, inputting the optimized execution tree to the evaluator, and communicating between the evaluator and any of a number of data reference devices. An execution tree with results is generated, the query results are packaged and the results displayed in an output device. (Col. 5, lines 1-5)

Although the Pouschine reference briefly alludes to the execution tree being optimized, the disclosure is very limited. Other than combining the SQL queries whenever possible, the Pouschine reference does not teach or suggest any method for process optimization. Nor does Pouschine disclose "evaluating processing options." Further, as conceded in the Office Action, Pouschine does not disclose "*an intermediate data processing method module for evaluating a plurality of methods for generating intermediate data sets within the data source(s).*" Office Action, page 8.

The Examiner cites Shwartz as allegedly disclosing "an intermediate data processing method module for evaluating a plurality of methods for generating intermediate data sets within the data source(s)." Office Action, pages 8-9. However, a close reading of the Shwartz reference reveals otherwise.

First, Shwartz does not deal with process optimization at all. The primary and only concern of Shwartz is to prohibit the selection of semantically incorrect query parameters. See

Title and Abstract of Shwartz. Therefore, Shwartz only distinguishes two types of queries, those that are semantically correct versus those that are semantically incorrect. Shwartz does not improve or optimize the efficiency of the query structure by evaluating multiple process options or multiple methods for generating intermediate data sets. Indeed, the text of the Shwartz reference does not even include such terms as "optimize," "evaluate," "efficient," or variations thereof.

Second, the intermediate query language referred to in Shwartz is completely different from the "intermediate data sets" as recited in claims 1, 10 and 21. Shwartz's intermediate query language is a translation of the user's query input from a form of menu selections to a form of easy-to-understand sentences. Col. 9, lines 45-47. In other words, Shwartz's intermediate query language is no more than an alternative presentation of the user input. In the present application, the "intermediate data sets" refer generally to data calculated or otherwise generated for the purpose of returning the desired data set. See page 15-17 for a description of the intermediate table selection module and the intermediate table method module. Unlike Shwartz's intermediate query language, the intermediate data sets in the present application are not user inputs, but data generated *in response to* user inputs.

In view of the foregoing, the § 103(a) rejections of the independent claims cannot stand.

B. The Rejection Of Claims 2, 5, 11 And 44 As Being Unpatentable Over Pouschine In View Of Leung Under 35 U.S.C. 103(a) Is Similarly Improper

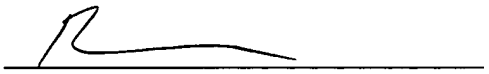
Because the rejections of the independent claims cannot stand, the basis is moot for rejecting dependent claims 2, 5, 11 and 14 based on Pouschine further in view of Leung. As such, the rejection of these dependent claims should be reversed and the claims should be allowed.

VIII. Conclusion

Because the cited references, taken either singly or in combination, fail to teach or suggest the combinations set forth in the pending claims, and further fail to provide any motivation or suggestion of the desirability of modifying the structures or methods to arrive at the claimed combinations, appellant submits that the pending claims are allowable over the cited references. Accordingly, appellant respectfully requests that the Board reverse the prior art rejections set forth in the Action, and allow all of the pending claims.

Respectfully submitted,

December 30, 2005



Brian M. Buroker
Registration No. 39,125

Hunton & Williams
1900 K. St., NW, Suite 1200
Washington, D.C. 20006-1109
(202) 955-1894

IX. APPENDIX A - Pending Claims

What is claimed is:

1. A computer-implemented system for retrieval and processing of a data set from one or more data sources comprising:

a query structure assembly module for defining a query structure based upon a plurality of query assembly rules and a desired data set, the query assembly rules being used by the query structure assembly module to evaluate the desired data set;

a syntax assembly module for defining at least one query language statement based upon the defined query structure;

a process optimization module for evaluating processing options based upon a database schema associated with the data source, the process optimization module including an intermediate data processing method module for evaluating a plurality of methods for generating intermediate data sets within the data source(s); and

whereby at least one query language statement is assembled and run against the data source(s) to return the desired data result set.

2. The system of claim 1, wherein the process optimization module includes a table selection module for evaluating the size of a selected set of tables from the database schema.

3. The system of claim 1, wherein the process optimization module includes an intermediate data processing selection module for evaluating the reusability of an intermediate data set in returning the defined data result set.

4. Cancelled.

5. The system of claim 1, wherein the process optimization module includes a join path selection module for evaluating the length of at least one join path used in returning the defined data result set.

6. The system of claim 1, wherein the query structure assembly module accesses a query assembly rule associated with a selected database management system.

7. The system of claim 1, wherein the syntax assembly module accesses a syntax description associated with a selected database management system.

8. The system of claim 1, wherein the process optimization module accesses a query assembly rule, a syntax description, or a syntax pattern associated with a selected database management system.

9. The system of claim 1, wherein the system is a component in an online analytical processing systems, a reporting system, a business intelligence system, or a data mining system.

10. A computer-implemented method of generating a query language statement to be run against one or more data sources, comprising the steps of:

generating a query structure based upon a database schema associated with the data source, query assembly rules, and a desired data result set, the query assembly rules being used to evaluate the desired data set;

generating query language syntax based upon the query structure for returning the desired data result set from the data source(s);

evaluating a plurality of query assembly rules, syntax descriptions, or syntax patterns for process optimization; and

evaluating a plurality of methods for generating intermediate data sets.

11. The method of claim 10, wherein the step of evaluating the plurality of query assembly rules, syntax descriptions, or syntax patterns for process optimization includes evaluating the size of a plurality of sets of identified tables for returning the desired data result set.

12. The method of claim 10, wherein the step of evaluating the plurality of query assembly rules, syntax descriptions, or syntax patterns for process optimization includes evaluating the reusability of intermediate data sets.

13. Cancelled.

14. The method of claim 10, wherein the step of evaluating a plurality of query assembly rules, syntax descriptions, or syntax patterns for process optimization includes evaluating a plurality of join paths used in returning the desired data result set.

15. The method of claim 10, wherein the step of evaluating a plurality of query assembly rules, syntax descriptions, or syntax patterns for process optimization includes evaluating at least one query assembly rule, syntax description, or syntax pattern associated with a selected database management system.

16. The method of claim 10, wherein the step of generating a query structure includes evaluating at least one query assembly rule associated with a selected database management system.

17. The method of claim 10, wherein the step of generating query language syntax includes evaluating at least one syntax description or syntax pattern associated with a selected database management system.

18. The method of claim 10, wherein the method is implemented in an online analytical processing systems, a reporting system, a business intelligence system, or a data mining system.

19. A medium having a processor readable program code embodied therein for retrieving and processing data from one or more data sources comprising:

code for causing the processor to evaluate a plurality of sets of tables within the data source(s) for generating a desired data result set;

code for causing the processor to evaluate at least one intermediate data set for reusability in generating the desired data result set;

code for causing the processor to evaluate a plurality of methods for generating intermediate data sets for use in generating the desired data result set;

code for causing the processor to evaluate a plurality of join paths used for joining tables to return the desired data result set; and

code for causing the processor to assemble at least one query language statement based upon the query structure and the evaluations of the plurality of sets of tables, the at least one intermediate data set, the plurality of methods for generating intermediate data sets, and the plurality of join paths.

20. The medium of claim 19, further comprising code for causing the processor to evaluate at least one query assembly rule associated with a selected database management system.

21. A computer-implemented system for constructing a structured query language statement to be run against at least one database, comprising:

a query structure assembly module for constructing a query structure based upon an evaluation of a desired data set by at least one query assembly rule;

a syntax assembly module for defining at least one query language statement based upon the constructed query structure; and

a process optimization module for evaluating the construction of the query structure and the defining of the at least one query language statement, the evaluation occurring during the construction of the query structure and/or during the defining of the at least one query statement, the process optimization module serving to evaluate a plurality of methods for generating intermediate data sets.

22-25. Cancelled.

26. A computer-implemented system for retrieval and processing of a data set from one or more data sources comprising:

a query structure assembly module for defining a query structure based upon a plurality of query assembly rules and a desired data set, the query assembly rules being used by the query structure assembly module to evaluate the desired data set;

a syntax assembly module for defining at least one query language statement based upon the defined query structure;

a process optimization module for evaluating processing options based upon a database schema associated with the data source, the process optimization module including an intermediate data processing method module for evaluating a plurality of methods for generating intermediate data sets within the data source(s);

whereby at least one query language statement is assembled and run against the data source(s) to return the desired data set; and

wherein the intermediate data processing method module determines whether creation of a permanent table, temporary table, view, derived table, or sub-query is the most efficient method for handling intermediate data calculations.

27. A computer-implemented method of generating a query language statement to be run against one or more data sources, comprising the steps of:

generating a query structure based upon a database schema associated with the data source, query assembly rules, and a desired data result set, the query assembly rules being used to evaluate the desired data set;

generating query language syntax based upon the query structure for returning the desired data result set from the data source(s);

evaluating a plurality of query assembly rules, syntax descriptions, or syntax patterns for process optimization;

evaluating a plurality of methods for generating intermediate data sets; and

wherein the step of evaluating a plurality of methods for generating intermediate data sets comprises determining whether creation of a permanent table, temporary table, view, derived table, or sub-query is the most efficient method for handling intermediate data calculations.

28. A medium having a processor readable program code embodied therein for retrieving and processing data from one or more data sources comprising:

code for causing the processor to evaluate a plurality of sets of tables within the data source(s) for generating a desired data result set;

code for causing the processor to evaluate at least one intermediate data set for reusability in generating the desired data result set;

code for causing the processor to evaluate a plurality of methods for generating intermediate data sets for use in generating the desired data result set;

code for causing the processor to evaluate a plurality of join paths used for joining tables to return the desired data result set;

code for causing the processor to assemble at least one query language statement based upon the query structure and the evaluations of the plurality of sets of tables, the at least one intermediate data set, the plurality of methods for generating intermediate data sets, and the plurality of join paths; and

code for determining whether creation of a permanent table, temporary table, view, derived table, or sub-query is the most efficient method for handling intermediate data calculations.

29. A computer-implemented system for constructing a structured query language statement to be run against at least one database, comprising:

a query structure assembly module for constructing a query structure based upon an evaluation of a desired data set by at least one query assembly rule;

a syntax assembly module for defining at least one query language statement based upon the constructed query structure;

a process optimization module for evaluating the construction of the query structure and the defining of the at least one query language statement, the evaluation occurring during the construction of the query structure and/or during the defining of the at least one query statement, the process optimization module serving to evaluate a plurality of methods for generating intermediate data sets; and

wherein the process optimization module's evaluation of a plurality of methods for generating intermediate data sets comprises determining whether creation of a permanent table, temporary table, view, derived table, or sub-query is the most efficient method for handling intermediate data calculations.